intel®

APPLICATION NOTE

AP-317

June 1988

# Implementing a PS/2 POS Using the 5AC312 EPLD

**PEDRO VARGAS**
PROGRAMMABLE LOGIC APPLICATIONS

Order Number: 292047-001

## INTRODUCTION

The introduction of the IBM* PS/2 (Personal System/2*) models and the innovative Micro Channel* has provided numerous opportunities to develop creative interface solutions. Although the interface requirements are new, the designer is faced with making a familiar choice: Use discrete chips (SSI/MSI), incorporate a PLD, or go for the custom IC solution.

In the past, using TTL on the PC/XT/AT bus was often a good choice, but the reduced size of the PS/2 adapters ("plug in boards") increases the cost of board space dramatically. The custom chip solution is probably the best for companies that have a well-defined product, large volumes, and can afford the cost of the chip development. The third choice, using a PLD, is one that has not been popular in PC bus interfacing due to the limited function and performance of most PLDs.

The Intel 5AC312 is a third-generation EPLD that gives designers the resources needed to interface to buses like the Micro Channel. In addition, it provides two benefits not completely provided by either of the other two choices; high integration, and re-programmability. The rest of this application note contains a detailed presentation of a basic POS (Programmable Option Select) implementation for the PS/2 Micro Channel that is done with the 5AC312 EPLD.

## PS/2 MICRO CHANNEL

One of the best features in the PS/2 models is the capability to do system and adapter configuration with software instead of hardware. This feature, called POS (Programmable Option Select), eliminates the need for switches on the motherboard and adapters by replacing them with programmable registers. The idea is, rather than removing boards and manually setting switches, all configuration information is located in files and can be read or written to the motherboard or to the adapters through the Micro Channel. The motherboard and each connector on the Micro Channel has a unique signal called -CD SETUP that initiates a setup mode when it is active. Only one connector at a time can be in the setup mode, which provides an organized way to perform initialization.

## POS REQUIREMENTS

Each adapter must implement POS with eight registers. Depending on the adapter function, not all of them need to be used. The first three (POS registers 0,1,2) are required because they provide the adapter ID and the adapter enable/disable function necessary during setup and error checking. In brief, the way that the system uses POS is as follows:

1. The system selects the adapter to be placed in setup mode by driving its -CD SETUP signal active.
2. The adapter is identified by reading two ID bytes from POS 0 and POS 1 (HEX 100 and 101).
3. The adapter is disabled by writing "0" to POS 2 (HEX 102).
4. If implemented, Option Select Data is written to POS 3, 4, 5.
5. The adapter is enabled by writing "1" to POS 2.
6. The adapter is out of setup mode when the system drives the -CD SETUP signal inactive.

The actual hardware implementation of POS is summarized in IBM technical documents, but the details are left up to each designer.

## ADAPTER REQUIREMENTS

The adapter used for this design is an Intel single-function card that incorporates two modems controlled by a 80C186. Since it performs only one function, there was no need to implement the POS Option Select bytes. (These POS bytes are used with multi- function adapters that do more than one task and reside in the system with similar adapters.) In this case, the only requirements were to provide the ID bytes and the enable/disable features, which are done with POS registers 0,1, and 2. Figure 1 shows the POS register layout and the typical POS hardware implementation as suggested by IBM. Table 1 defines the POS registers.

*IBM, Personal System/2 and Micro Channel are trademarks of International Business Machines Corporation.

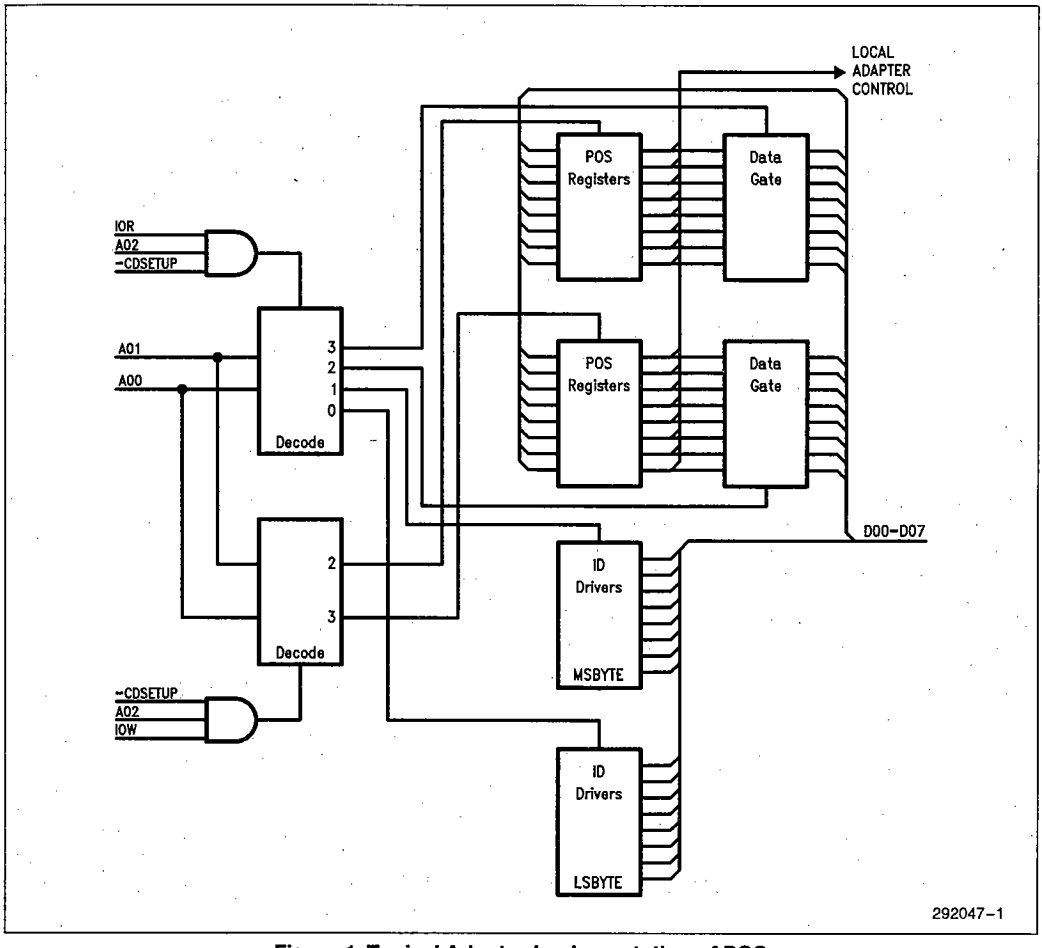



**Figure 1. Typical Adaptor Implementation of POS**

**Table 1. POS I/O Address Decode**

| Address (hex) | Register | -CD SETUP | Address Bit | | | Function |
|---|---|---|---|---|---|---|
| | | | A2 | A1 | A0 | |
| 0100 | POS Register 0 | 0 | 0 | 0 | 0 | Adapter Identification Byte (Least Significant Byte) |
| 0101 | POS Register 1 | 0 | 0 | 0 | 1 | Adapter Identification Byte (Most Significant Byte) |
| 0102 | POS Register 2 | 0 | 0 | 1 | 0 | Option Select Data (Byte 1)* |
| 0103 | POS Register 3 | 0 | 0 | 1 | 1 | Option Select Data (Byte 2) |
| 0104 | POS Register 4 | 0 | 1 | 0 | 0 | Option Select Data (Byte 3) |
| 0105 | POS Register 5 | 0 | 1 | 0 | 1 | Option Select Data (Byte 4)* |
| 0106 | POS Register 6 | 0 | 1 | 1 | 0 | Subaddress Extension (Least Significant Byte) |
| 0107 | POS Register 7 | 0 | 1 | 1 | 1 | Subaddress Extension (Most Significant Byte) |

*These bytes contain one or more bits with specific value assignments

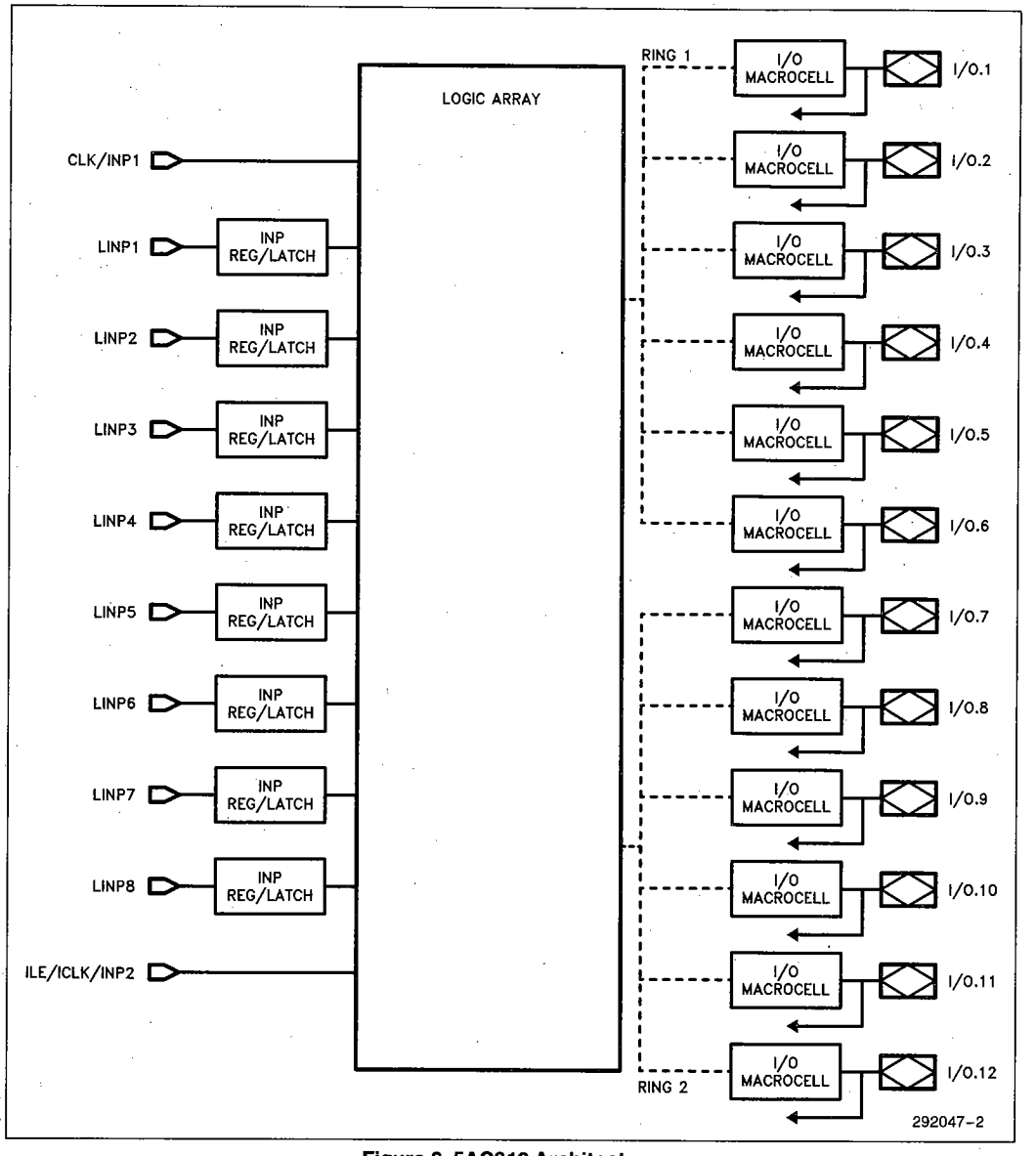



**Figure 2. 5AC312 Architecture**

## 5AC312 EPLD

With 12 macrocells and a host of other features, the 5AC312 is Intel's newest EPLD. The device is based on the same CHMOS process used in other Intel devices. This EPLD provides an abundant feature set, but its strength lies in being able to efficiently implement one very important function missing from most PLDs: register-logic-register functions.

## DEVICE DESCRIPTION

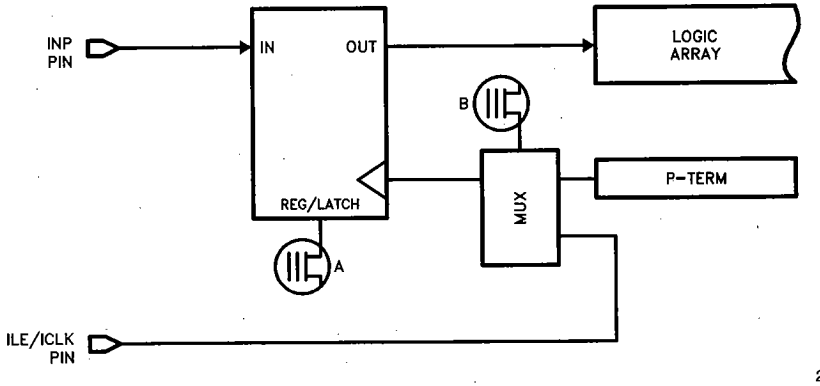
The 5AC312 (Figure 2) contains 12 macrocells with programmable outputs and inputs. A macrocell is the basic block associated with each output register within the EPLD. The 5AC312 has the following features:

- 12 I/O macrocells with dual feedback for implementing buried registers.

- 8 programmable inputs that can be configured as latches, registers, or flow through inputs. These can be clocked synchronously or asynchronously.
- Product term allocation on each macrocell.
- 2 product terms on all macrocell control signals.
- 2 multi-function pins; a CLK/INPUT and a ILE/ICLK/INPUT.
- 40 MHz operation.

The 5AC312 provides three major benefits that are especially important to designers working on bus interfaces:

1. The availability of input latches (Figure 3) makes it easy to synchronize bus control signals synchronously or asynchronously. The latches can be clocked as a group of 8 or individually, as is quite common on most buses. Input latches also make state machine designs more reliable. Since buses are prone to glitches and other transients, the ability to hold the inputs stable while transitioning through states makes the difference between a clean and a jittery state machine.
2. Product Term Allocation (Patent Pending) brings a new concept to the Intel EPLD family and makes the 5AC312 unique among PLDs. This feature means that the designer can implement large designs that contain as few as zero or as many as 16 product terms per macrocell (Figure 4). Product Term Allocation takes place in two rings of six macrocells. Within each ring (Figure 2), individual macrocells can allocate p-terms to/from adjacent macrocells. This is a real benefit in bus decoding where intermediate signals can have few or many p-terms all within the same logic function. Most designers that use PLDs have at least one horror story of a design that required 10 or more p-terms and a device that could only provide 8.
3. A flexible output structure is a must for efficient bus interfacing, which quite commonly requires lots of I/O and complex control signals. The 5AC312 meets these demands head-on with dual-feedback paths and two p-terms per control signal on all I/O macrocells. This means that certain functions, like state machines, can be buried and a pin won't be wasted because it can be used as an additional input. Also, output enables and register operations are frequently generated by a combination of memory, I/O, read, and write strobes. Many times these control signals require two p-terms or the equivalent of an external read/write multiplexer. Prior to the 5AC312, the only way to implement this in PLDs was to waste a macrocell to inefficiently provide this function. Figure 5 shows the macrocell structure and details this third benefit.




**NOTE:**
Flow-through input selected by connecting ILE P-Term to $V_{CC}$.

**Figure 3. 5AC312 Input Structure**

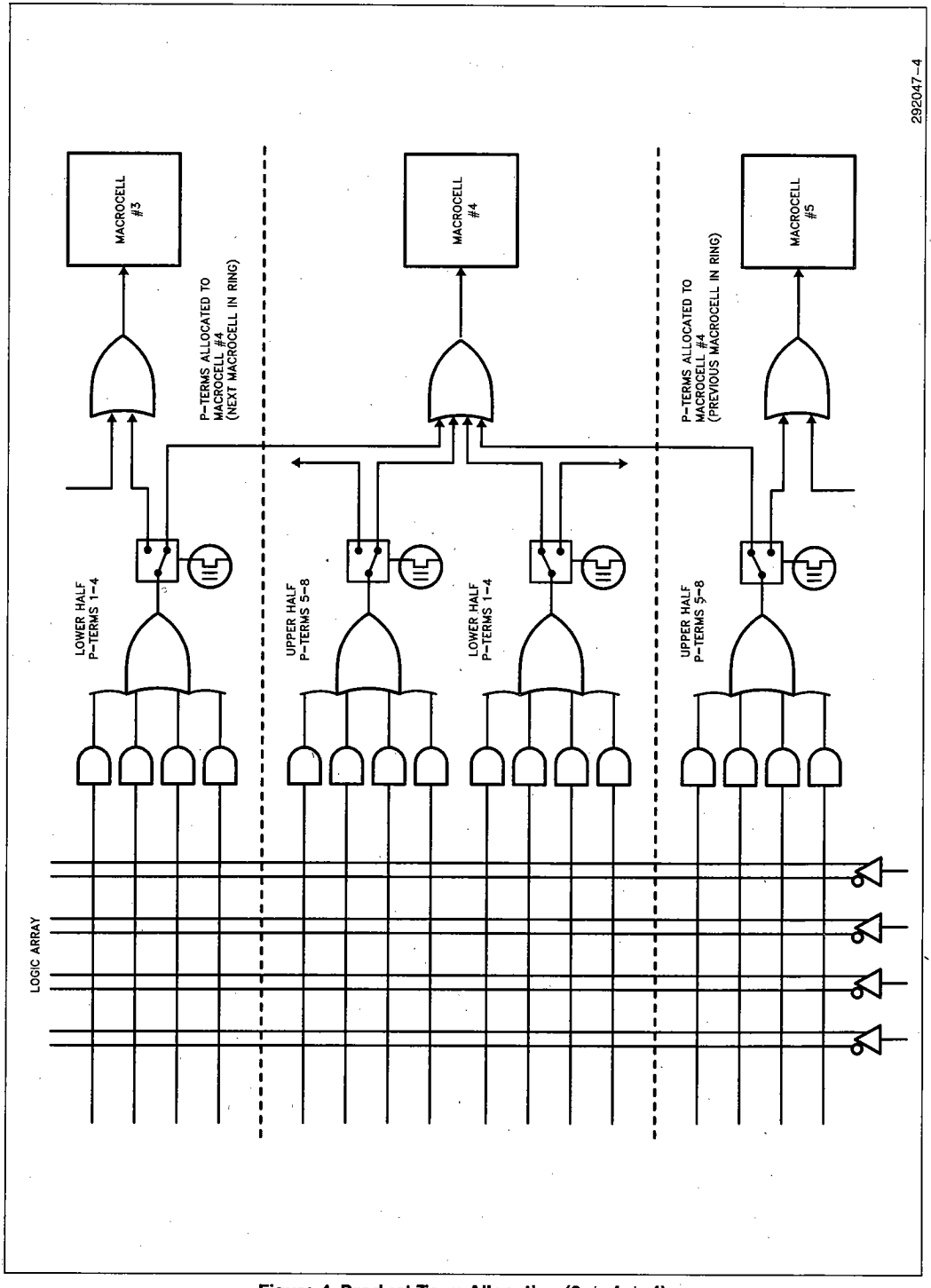



**Figure 4. Product Term Allocation (8 + 4 + 4)**

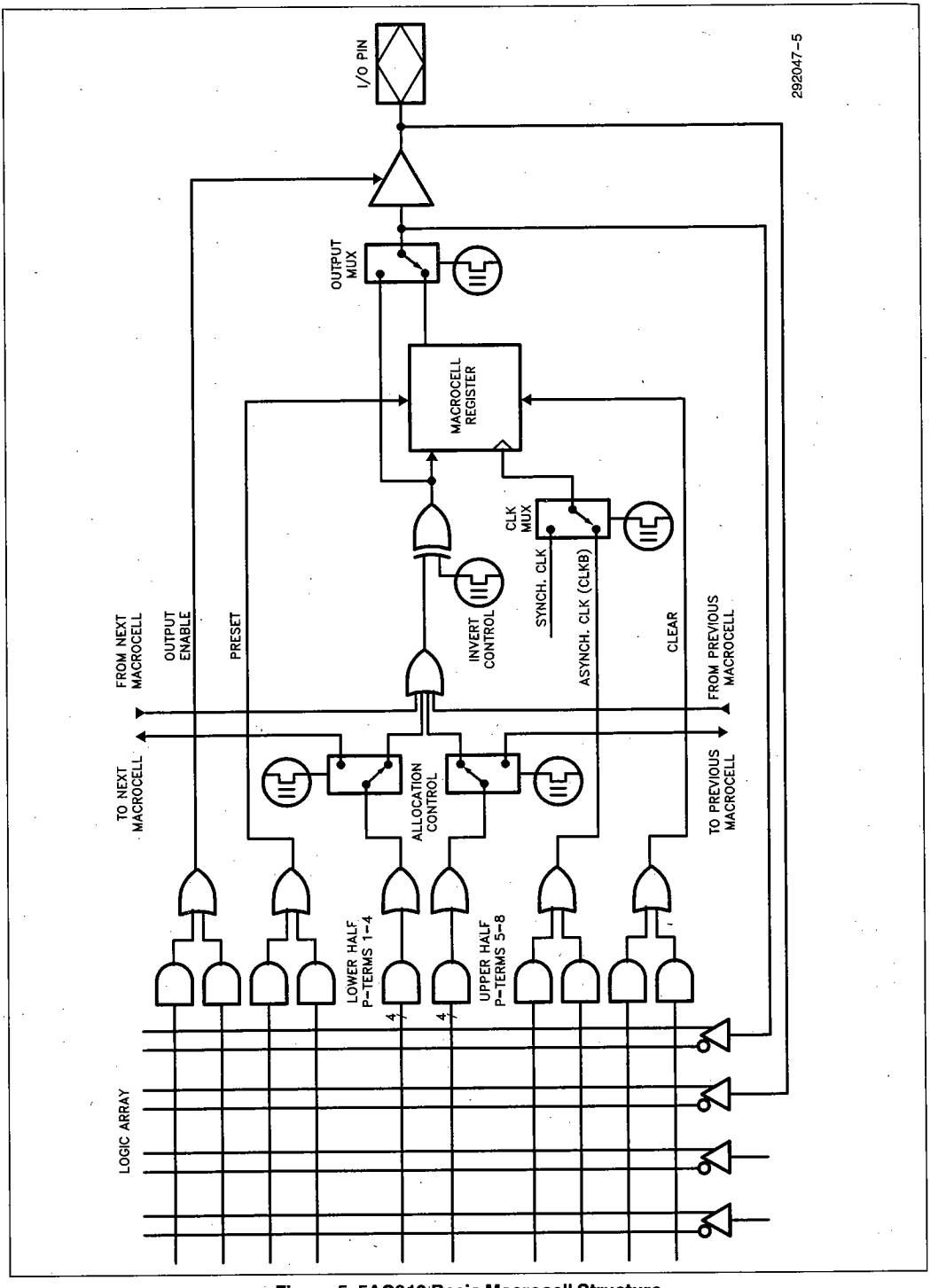



**Figure 5. 5AC312 Basic Macrocell Structure**

Since it is a CMOS EPLD, the 5AC312 has very frugal impact on the current allotted by the PS/2 power supplies. The device consumes much less power than an equivalent TTL or PAL implementation.

## 5AC312 POS IMPLEMENTATION

The POS requirements placed on our modem adapter are easily met by putting the 5AC312 to some creative use. In terms of performance, the 25 ns propagation delay and capability to operate to 40MHz is more than adequate for the system requirements during setup mode.

## RESOURCE ALLOCATION

POS registers 0, 1, and 2 can be easily accommodated with 12 macrocells. Eight macrocells are used to load and output the ID bytes from POS 0 and 1. One macrocell is used as the LSB of POS register 2. The remaining three macrocells make up a state machine that internally sequences through the setup mode. The partitioning used for this design is shown in Figure 6.

The state diagram shown in Figure 7 explains the operation of the design. During S0 (state 0), the 5AC312 idles until -CD SETUP is driven active. The adapter is already disabled and POS 2 is zero because during power-up and reset the 5AC312 registers come up as logic 0. When -CD SETUP is driven active, the 5AC312 goes to S1 and loads the first ID byte, FFH. It remains in S1 while waiting for a READ POS 0 command. As soon as POS 0 is read the state machine cycles to S2 and outputs FFH which is the least significant byte for our adapter. Once READ POS 0 is inactive, the 5AC312 cycles to S3 where it loads the second ID byte (7FH) and waits for READ POS 1 active. The last ID byte is put on the bus when READ POS 1 comes and the machine goes to S4. Since we know that the next two setup operations are I/O writes, the 5AC312 remains in S5 while POS 2 is disabled and enabled per the Micro Channel bus specification. This operation, which is a bit write of a register, is easily done by using the 5AC312's dual feedback capability. While bit 0 of POS 2 uses the D00 line, internally it gets routed to a separate register. Without dual feedback this internal transceiver function would be impossible to implement.

The IBM Technical Reference Manual provides a table for suggested ID bytes arranged by adapter type. The modem card falls under the category of storage device, so 7FFFH was chosen. While IBM has assigned unique IDs for its own cards the third party choices are up

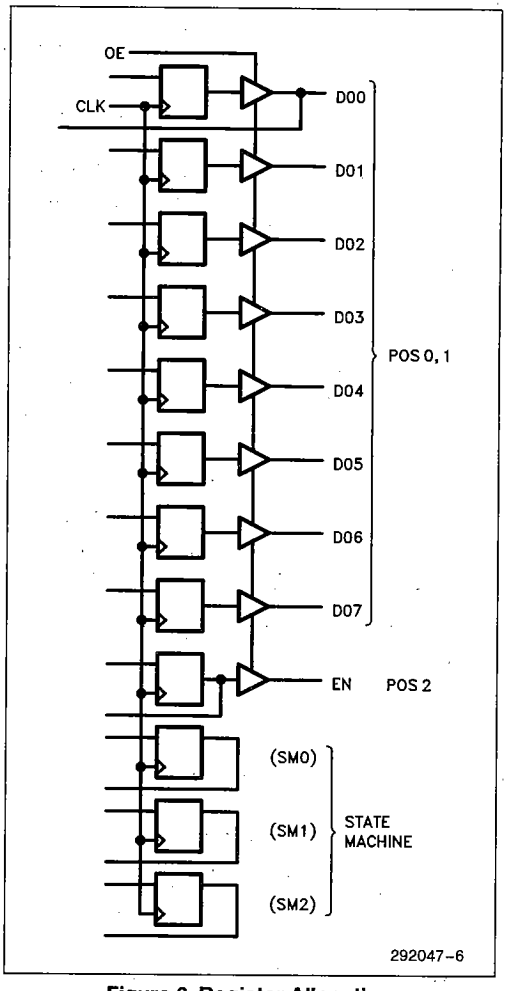



**Figure 6. Register Allocation**

for grabs. It is conceivable that more than one company may assign the same ID to their own cards. In this case, companies that implement the POS function in discrete TTL or a custom IC may have a problem. That is, they'll either have to re-do the design, which could be expensive, or, cut and jumper the board, which, goes against the Micro Channel specification and still costs.

Since the 5AC312 is re-programmable, the risk of conflicting IDs is minimized since all that is needed

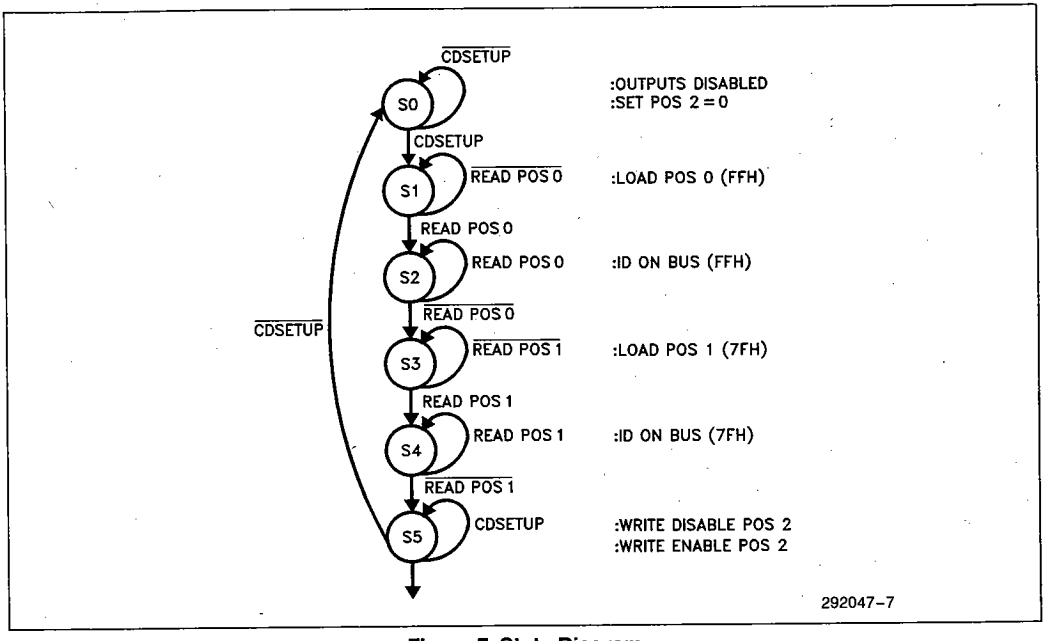



**Figure 7. State Diagram**

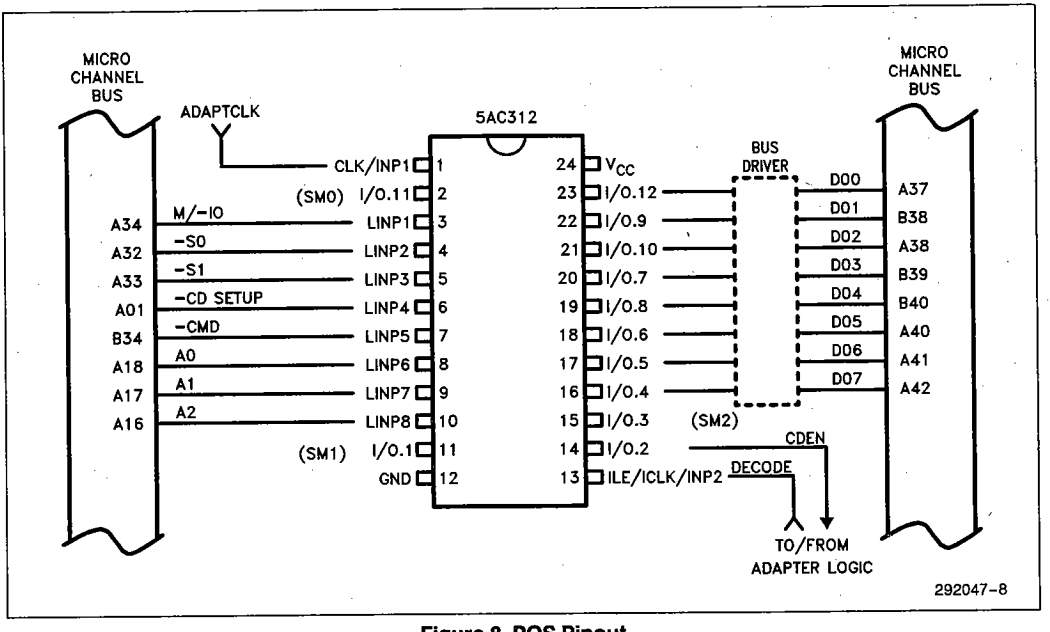



**Figure 8. POS Pinout**

is to burn another EPLD with the appropriate bytes. The pinout for the 5AC312 POS implementation is shown in Figure 8. Note that a bus driver is required to support the current levels on the Micro Channel. The definition for the signals can be found in the IBM literature, but briefly is as follows:

| | |
|---|---|
| M/-I/O | Memory or Input/Output. |
| -S0, -S1 | Status bits 0 and 1. |
| -CD SETUP | Card Setup. |
| -CMD | Command. |
| A0–A3 | Address bits 0–3. |
| D00–D07 | Eight bit data bus. |
| DECODE | Adapter decode for the higher order 16 bit address bus. |
| EN/-DIS | Enable/Disable adapter. |

## DESIGN FILES

The 5AC312 was developed and programmed using the Intel IPLSII 1.5 EPLD software. This software provides a variety of entry methods like Boolean equation, state machine, and schematic capture. For this particular design, equation entry was the most convenient since the implementation was done in one IC.

The minimized and ordered .LEF (Logic Equation File) for the 5AC312 is shown in Figure 9. A quick glance at a few items really points out the power of the 5AC312. For example, the equation for variable ENd has 11 p-terms. Without p-term allocation this would not have fit unless it was done with two macrocells, which is wasteful. Also, the output enable control signal OE contains two p-terms for each macrocell. Again, without the 5AC312 some work-around may have been possible. But its unlikely that the design could have fit in one device, resulting in a functional but not too optimal solution. Finally, in keeping with good state machine design practices, all of the critical bus signals were received by registers with the EPLD primitive RINP (Registered Input).

A description of the Micro Channel Adapter Description File and the Configuration Utilities is beyond the scope of this article. The IBM literature is very descriptive in how adapters are setup and configured, and the reference section contains the names of the pertinent documents.

This modem adapter POS implementation was easily done in one 5AC312 device. Adding other Micro Channel interface features like arbitration or interrupt sharing would overburden it. Since this adapter did not have those requirements it was not a problem. However, a complete POS implementation with Option Select Bytes and other features could be done with the 5AC312's big brother, the 5AC324.

## SUMMARY

Interfacing to the PS/2 Micro Channel can be a difficult chore when using PLDs or other solutions that are not flexible or powerful enough. However, the 5AC312 with its powerful features like product term allocation is a giant step in the right direction to making the job easier.

## ACKNOWLEDGEMENTS

Many thanks to Thom Bowns and Dan Smith for their help with this article.

## REFERENCES

1. IBM Personal System/2, Model 80, Technical Reference.
2. IBM Personal System/2, Hardware Maintenance Reference.
3. Intel 5AC312 EPLD Data Sheet.
4. *Electronics* "Inside Technology", September 17 1987.

```
THOM BOWNS
INTEL
DECEMBER 4, 1987
1
001
5AC312
Implements POS for the PS/2 using a 5AC312.
LEF Version 1.5 Baseline 4.1i 21 Nov 1987
OPTIONS: TURBO=ON
PART:
      5AC312
INPUTS:
      MIO@3, nS0@4, nS1@5, nCDSETUP@6, nCMD@7, A0@8, A1@9, A2@10, DECODE@13,
      CLK@1, D0@23
OUTPUTS:
      D00@23, D01@22, D02@21, D03@20, D04@19, D05@18, D06@17, D07@16, EN@14,
      SM0@2, SM1@11, SM2@15
NETWORK:
      IRE = CLKB(CLK)
      CLK = INP(CLK)
      MIO = RINP(MIO, IRE, GND, GND)
      nS0 = RINP(nS0, IRE, GND, GND)
      nS1 = RINP(nS1, IRE, GND, GND)
      nCDSETUP = RINP(nCDSETUP, IRE, GND, GND)
      nCMD = RINP(nCMD, IRE, GND, GND)
      A0 = RINP(A0, IRE, GND, GND)
      A1 = RINP(A1, IRE, GND, GND)
      A2 = RINP(A2, IRE, GND, GND)
      DECODE = INP(DECODE)
      D0 = INP(D0)
      D00 = RONF(D00d, CLK, GND, GND, OE)
      D01 = RONF(D01d, CLK, GND, GND, OE)
      D02 = RONF(D02d, CLK, GND, GND, OE)
      D03 = RONF(D03d, CLK, GND, GND, OE)
      D04 = RONF(D04d, CLK, GND, GND, OE)
      D05 = RONF(D05d, CLK, GND, GND, OE)
      D06 = RONF(D06d, CLK, GND, GND, OE)
      D07 = RONF(D07d, CLK, GND, GND, OE)
      EN, EN = RORF(ENd, CLK, GND, GND, VCC)
      SM0 = NORF(SM0d, CLK, GND, GND)
      SM1 = NORF(SM1d, CLK, GND, GND)
      SM2 = NORF(SM2d, CLK, GND, GND)
```

292047-9

**Figure 9. POS Design File**

```
EQUATIONS:
      SM2d = SM2' * SM1 * SM0 * nCDSETUP' * MIO' * nS0' * A2' * A1' * DECODE *
             nS1
           + SM2 * SM1' * nCDSETUP';

      SM1d = SM2' * SM1' * SM0 * nCDSETUP' * MIO' * nS0' * A2' * A1' * DECODE *
             nS1
           + SM2' * SM1 * SM0' * nCDSETUP'
           + SM2' * SM1 * nCDSETUP' * DECODE'
           + SM2' * SM1 * nCDSETUP' * A1
           + SM2' * SM1 * nCDSETUP' * A2
           + SM2' * SM1 * nCDSETUP' * nS1'
           + SM2' * SM1 * nCDSETUP' * nS0
           + SM2' * SM1 * nCDSETUP' * MIO;

      SM0d = (SM2' * SM0 * MIO' * nS0' * nS1 * A2' * A1' * DECODE
           +  SM2 * SM0' * MIO' * nS0' * nS1 * A2' * A1' * DECODE
           +  SM1 * MIO' * nS0' * nS1 * A2' * A1' * DECODE
           +  SM2 * SM1
           +  nCDSETUP)';

      ENd = D0 * MIO' * A2' * A1 * A0' * DECODE * SM2 * SM1' * SM0 * nS1' * nS0
          + nS0' * EN
          + nS1 * EN
          + SM0' * EN
          + SM1 * EN
          + SM2' * EN
          + DECODE' * EN
          + A0 * EN
          + A1' * EN
          + A2 * EN
          + MIO * EN;

      D07d = SM2' * SM0'
           + SM2' * SM1';

      D06d = (SM2 * SM1)';

      D05d = (SM2 * SM1)';

      D04d = (SM2 * SM1)';

      D03d = (SM2 * SM1)';

      D02d = (SM2 * SM1)';

      D01d = (SM2 * SM1)';

      OE = SM2 * SM1' * SM0' * MIO' * nS0' * A2' * A1' * DECODE * nS1
         + SM2' * SM1 * SM0' * MIO' * nS0' * A2' * A1' * DECODE * nS1;

      D00d = (SM2 * SM1)';

END$
```

292047-10

**Figure 9. POS Design File** (Continued)